Naval Research Laboratory

Washington, DC 20375-5320

NRL/MR/5540--09-9191

A Framework for Automatic Web Service Composition

ANYA KIM
MYONG KANG
CATHERINE MEADOWS

Center for High Assurance Computer Systems
Information Technology Division

ELIAS IOUP
JOHN SAMPLE

Mapping, Charting, and Geodesy Branch
Marine Geosciences Division

April 30, 2009

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
30-04-2009	Memorandum Report	February 2009 – April 2009

4. TITLE AND SUBTITLE

A Framework for Automatic Web Service Composition

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER
0602435N

6. AUTHOR(S)

Anya Kim, Myong Kang, Elias Ioup, Catherine Meadows, and John Sample

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Center for High Assurance Computer Systems
4555 Overlook Avenue, SW
Washington, DC 20375-5320

Naval Research Laboratory
Marine Geosciences Division
Stennis Space Center, MS 39529

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/5540--09-9191

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
One Liberty Center
875 North Randolph Street
Arlington, VA 22203-1995

10. SPONSOR / MONITOR'S ACRONYM(S)

ONR

11. SPONSOR / MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Complex mission plans may need to incorporate information from various sources and domains to achieve a task. This information is available through a variety of web services in the Service-Oriented Architecture (SOA), but the ability to automatically compose them into a single coherent task is not readily available. Traditional composition approaches require human-intensive involvement, making them time-consuming and error prone. Therefore, the ability to automatically or semi-automatically orchestrate web services in a short timeframe is highly desirable.

Recent works in the area of automatic web service composition produced a plethora of automation approaches with different degrees of automation. Different situations call for different composition approaches. However, there is a lack of guidance regarding what approaches are appropriate for a particular situation. In this paper, we examine the various approaches and develop a general-purpose framework for automatic service composition. Within the framework, we outline the common steps in the various composition processes and review the options available at each step. We also provide guidelines for choosing a composition approach within the framework for the geospatial planning domain.

15. SUBJECT TERMS
Web service composition
Service-oriented architecture
Geospatial project planning

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Anya Kim
Unclassified	Unclassified	Unclassified	UL	21	**19b. TELEPHONE NUMBER** *(include area code)* (202) 767-6698

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39.18

A Framework for Automatic Web Service Composition

Anya Kim[1], Myong Kang[1], Elias Ioup[2], Catherine Meadows[1], and John Sample[2]

[1]Center for High Assurance Computer Systems
Naval Research Laboratory
Washington DC 20375
{kim, mkang, meadows}@itd.nrl.navy.mil

[2]Marine Geosciences Division
Naval Research Laboratory
Stennis Space Center, MS 39529
{eioup, jsample}@nrlssc.navy.mil

Abstract

Complex mission plans may need to incorporate information from various sources and domains to achieve a task. This information is available through a variety of web services in the Service-Oriented Architecture (SOA), but the ability to automatically compose them into a single coherent task is not readily available. Traditional composition approaches require human-intensive involvement, making them time-consuming and error prone. Therefore, the ability to automatically or semi-automatically orchestrate web services in a short timeframe is highly desirable.

Recent works in the area of automatic web service composition produced a plethora of automation approaches with different degrees of automation. Different situations call for different composition approaches. However, there is a lack of guidance regarding what approaches are appropriate for a particular situation. In this paper, we examine the various approaches and develop a general-purpose framework for automatic service composition. Within the framework, we outline the common steps in the various composition processes and review the options available at each step. We also provide guidelines for choosing a composition approach within the framework for the geospatial planning domain.

1. Introduction

An atomic web service is a basic unit of operation in a Service-oriented Architecture (SOA). However, in general, an atomic web service is not able to satisfy the functional requirements of complex tasks. Therefore, it is desirable to logically connect several atomic web services to satisfy complex functional requirements, leveraging the loose coupling characteristics of SOA. For example, an atomic web service may be sufficient to retrieve a map of a location or obtain the weather conditions of a region. However, such piecemeal information by itself is not very useful. For example, in geospatial domain, one example application is the construction of a landing plan that takes location, time and

Manuscript approved April 15, 2009.

This work was sponsored by the U.S. Naval Research Laboratory's Base Program, Program Element No. 0602435N

equipment as inputs, and provides a map of the landing area annotated with weather, meteorology, and tidal conditions. Figure 1 shows a high-level abstract specification of such task with the circle and its label denoting the high level task and its goal, and inputs and outputs denoted as arcs.

Figure 1. Abstract specification of a landing plan task

In general, we would not expect to find an atomic web service that would take these exact inputs, produce the desired output, and satisfy any user constraints outlined in the task description. Therefore, we would need to take individual web services and put them together in a way that the goal of the landing plan is met. This requires decomposing an abstract specification into an abstract (composite) workflow composed of subtasks that eventually correspond to web services, with the subtasks connected through some process logic. For example, Figure 2 can be thought of as an abstract workflow derived from the high-level abstract specification of Figure 1. The circles represent the individual tasks, and the arcs denote the data flow. When each task is sufficiently simplified into subtasks by decomposing a higher-level specification and constraints, atomic web services are bound to these individual tasks, and executed according to the logic of the workflow.

The logic involved in creating a landing plan, or any other complex task for that matter, is not trivial. The workflow, or a series of tasks to create a successful landing plan, requires multiple tasks to fork and merge at appropriate steps. Other scenarios may be more complex, requiring conditional invocation, parallel execution of web services, etc.

Traditionally, web service composition has been performed manually, making it a difficult and error prone task. In this paper, we present a framework for automatic web service composition. We also discuss various options at each stage of the composition, and their pros and cons. Lastly, we provide a guideline for the geospatial domain as to what options to choose in each stage of the composition process, as well as a recommendation.

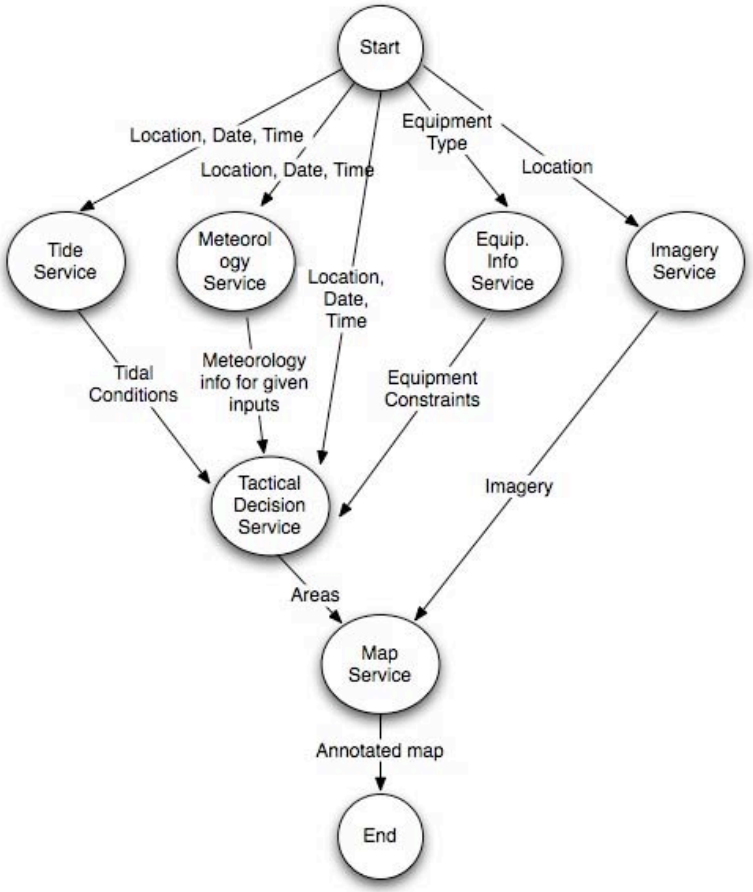

Figure 2. Abstract workflow of the high-level landing plan

2. Issues in Web Service Composition

In this section, we examine various steps of web service composition and review the state of art in automatic web service composition.

2.1. A Typical Manual Web Service Composition Process

In this section, we discuss the various steps for creating a composite web service manually. Throughout this paper, we use the landing plan that was introduced in section 1 as an example.

Initially, a user would create an abstract specification of a high-level task. The specification may include the task goals, I/O and control parameters, and any conditions and effects. As a next step, if the user possesses enough domain knowledge, he could create an abstract composite workflow similar to that of Figure 2 by decomposing the specification. If the user lacks domain knowledge, he would not have the skill set required to do this manually and therefore would not be able to create a composite workflow at all.

To create the abstract workflow, the user may utilize some web service standard language such as WS-BPEL [1] or OWL-S [2]. Besides the basic flow of the process, additional

constraints such as QoS parameters and security policy information may be specified for the task as a whole or each subtask within the workflow. At this point, the user needs to have some guarantee that the abstract workflow he created satisfies his intended high-level goal. Next, the user would have to bind web services to his workflow. The most widely used method is to search and discover services published in a web service repository such as UDDI [3]. There is no guarantee that services will be a one-to-one match with the task descriptions: I/O parameter differences may make them incompatible, several tasks may be fulfilled by one service, requiring further decomposition of the specification, or visa versa. Therefore, web service discovery is a non-trivial task. Once the user binds services to tasks, an executable process is created. As the process is executed, the user needs to monitor the execution and handle faults if they arise.

This kind of direct manipulation provides the user with a great deal of control over exactly how and what to specify. It also means that the user must possess a good deal of knowledge not only in the application domain but also about web service standards and security requirements, etc. It is also a time-consuming and error-prone process with no guarantee that the end result will indeed satisfy the user's requirements. Therefore, some tools and techniques that facilitate automatic composition are desirable.

2.2. Current State of the Art

Automation of web service composition is not a new concept. There are many proposed solutions for automated or semi-automated web service composition [4-10]. However, while a great deal of work has been done in this area, these works have mostly focused on only one or two aspects of the composition problem. A bulk of the literature focuses on the planning aspect of web service composition, with some attention to the validation and execution aspects. Papers that address the planning phase generally are divided into two categories; those that view planning as a workflow problem using templates [11-13], or as an AI planning problem [6, 7, 14].

Proposed solutions also tend to champion only one option or technology in the overall composition process. For example, there are papers that focus on fully automatic composition, while others propose semi-automatic (or user-involved) composition in various stages of service composition to ensure the validity of the composed workflow [15, 16]. Another area where authors' opinions diverge is in the use of semantics. While many focus on the syntactic representation of web service composition, others argue that without semantics, various composition strategies are not adequate enough to create workflows that satisfy user goals [17, 18].

When papers do address overall composition issues, they lack a high-level view, leaving the user mired in the low-level details and without enough options. For example, Rao and Su [8] present the following phases of the web service composition process: presentation of a single service, translation of the languages, generation of composition process model, evaluation of composite service, and execution of composite service. However, while they define these phases, they focus on the phase of web service process generation, and different methods available for that phase. They also do not discuss details of the pros and cons of the various options for each phase.

Finally, there has been some work in the geospatial domain related to web service composition [10, 19, 20]. However, these works suffer the same limitations that exist in the general web service composition literature. They either do not provide a useful framework [10, 19] or focus on the use of a set of technologies as a solution [20].

In short, recent literature on web composition provides too narrow a scope. Some only discuss a portion of the process in detail while others provide an overall view, but with a narrow perspective. We need a high-level web service composition framework that addresses the composition process from start to finish while providing the reader with alternatives/options at each step and discussing the pros and cons of these options.

2.3. Requirements for Automatic Web Service Composition

Having examined the process of a user creating a composite web service manually and looked at current solutions in the literature, we now propose our framework for automatic web service composition. We begin by defining five phases of the composition process, based on the steps outlined in section 2.1. They are specification, planning, validation, discovery and execution with monitoring phases. This is a high-level abstraction of the composition process, with no consideration to a particular platform, algorithm, or approach. Within each phase, we can derive requirements that need to be met in an automatic web service composition framework:

- Specification phase: Provide an easy way for a user to specify task goals, requirements and constraints without extensive domain knowledge
- Planning phase: Provide an automatic way to compose an abstract workflow based on the specification
- Validation phase: Provide techniques to ensure that the composite process realized via an abstract workflow satisfies the user's stated task goals
- Discovery phase: Provide a way to discover services that satisfy task specifications in the workflow
- Execution with monitoring phase: Provide a framework for monitoring an executing service, and provide automatic fault-handling mechanisms

Some of the infrastructure for this already exists (such as a web service repository). Also ontologies for semantic annotation are available albeit not used on a wide scale. There are available algorithms and tools for planning that assist the user to free them from the low-level technical details. These various components have to be put together in a cohesive way in order for the user to fully exploit them.

Figure 3 shows our composition framework annotated with inputs and outputs of each stage of the composition process.

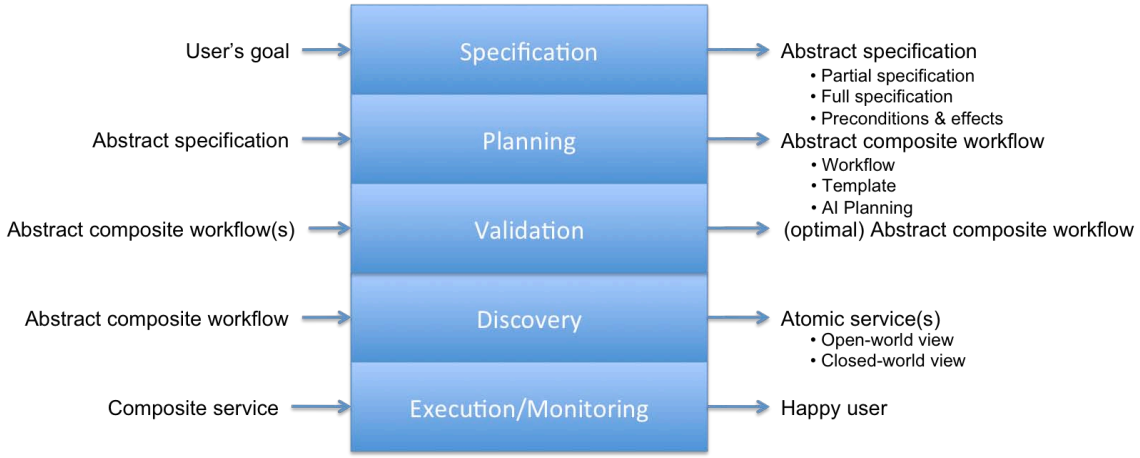

Figure 3. A proposed web service composition framework with respective inputs and outputs

3. A Web Service Composition Framework

This section presents a detailed description of the five composition phases in our proposed web service composition framework along with possible options.

3.1. Specification Phase

In this phase, the user specifies the goal he is trying to achieve from creating a complex task to be achieved through the composition of web services and produces an abstract specification. The specification should have enough detail to aid in creating the abstract specification and include functional and non-functional requirements. Functional requirements are constraints, control/data flow and high-level goal of the complex task. Non-functional aspects may include security policy, QoS constraints, etc. Desired behavioral issues (i.e. termination conditions, failure recovery requirements) should also be able to be specified.

3.1.1. Approaches

Tasks can be described in terms of the goal that they are trying to achieve and I/O parameters. For example, a credit card payment scheme can be described as having the goal of processing payments via credit cards, and taking a credit card number as input and receipt as output. In this approach there can be different levels of specificity such as a partial specification that outlines some of the tasks (or just a high level task such as that of Figure 1) or a full specification that details all the subtasks of the composition (such as that of Figure 2), depending on the user's level of domain knowledge. If a partial specification approach is used, the specification will need to be decomposed into smaller subtasks at a later time in order to create an abstract workflow.

Alternatively, when the process model is yet undefined or unknown, but the user does have a set of constraints and preferences, tasks can be described by their preconditions and effects (P/E) which effectively are the changes in state. Using the same example for a

credit card payment/processing service, the precondition would be that the user's credit card has sufficient funds, and the effects (i.e. goal states) would be that the card has been charged. This latter approach is used in many AI planning schemes and does not require the knowledge of a predefined workflow.

3.1.2. Tools and Components

The languages used to specify the abstract specification can range from using a WS standard language such as OWL-S profile or WS-BPEL, or a dedicated formal service specification language [21], 1st order logic, graph modeling languages, etc .

3.2. Planning Phase

The planning phase takes the abstract specification created in the specification phase and produces an abstract composite workflow. If the specification did not contain enough detail (i.e., it was a partial specification) or was stated as P/E and goals, then the description needs to be decomposed into simpler steps in this phase.

3.2.1. Approaches

There are three main approaches in this phase.

- Workflow-based: In workflow-based composition, a composite process is viewed as a workflow, depicted as an acyclic directed graph, with control and dataflow specified. If in the specification phase, a full specification was created using OWL-S, then that can be used as the abstract workflow for the planning phase. Workflow-based composition requires extensive domain knowledge and at least some degree of manual implementation by developers, making it difficult, time consuming and error-prone [22].
- Template-based: Templates describe the outline of activities needed to solve a problem. The templates can be parameterized with respect to some variables to allow customization based on user needs and preferences [13]. Templates are instantiated to create an executable workflow. The user can add security policies and requirements during the planning phase. Many template-based approaches extend standards such as OWL-S to describe templates [9, 11, 13, 23]. Once generated, templates have the advantage of being reused and extendible.
- AI Planning: AI planning techniques such as markov chains, backward chaining, graph theory-based, etc view web services as being described by their preconditions and effects. "AI planning is a way to generate a process automatically based on the specification of a problem. Planners typically use techniques such as progression (or forward state search), regression (or backward state search), and partial ordering." [10]. Most composition methods in this category are automatic and do not require user knowledge of predefined workflow.

3.2.2. Tools and Components

Most available planning tools use WS-BPEL with WSDL descriptions or OWL-S, while some AI planning tools use proprietary languages (such as PDDL, CSSL [8], etc) that provide easy to validate logic.

3.3. Validation Phase

The validation phase takes an abstract composite workflow that is an output of the planning phase to address the following:

- Syntactic validation: Is the workflow well-formed and structurally correct (i.e. does not contain deadlocks, infinite cycles, etc)? Syntactic validation may be handled by the planning tool, depending on the approach used.
- Semantic validation: Does the plan satisfy user goals and requirements/constraints that were detailed in the specification phase?

If the user created more than one abstract composite workflow in the planning phase, there may be multiple candidates to validate. The result of the validation should produce a syntactically and semantically optimal abstract composite workflow.

3.3.1. Approaches

Most validation algorithms that check for satisfaction of constraints (semantic validation) are based on utility theory. Some algorithms are based on shortest path heuristics [20] or model checking [24]. Assumptions are made about the complex task, and the subtasks within the workflow. These assumptions and properties that the workflow is expected to fulfill are modeled in a formal language. If more than one workflow satisfies the requirements, the most optimal one is chosen. The methods of ranking the workflow are mostly based on QoS (e.g. shortest execution time) and best matching of I/O parameters.

The validation process may be an iterative one requiring the user to going back to either the specification or planning phase to create a better matching workflow, if workflows in the candidate list do not satisfy the user's stated goals.

3.3.2. Tools and Components

Since WS-BPEL and OWL-S are the two main specification languages used to create orchestrations, the majority of tools are developed specifically for one or the other. For BPEL-created workflows, tools such as Eclipse's BPEL Project [25] and Oracle's BPEL Process Manager [26] provide authoring and deployment features as well as validation. For workflows created using OWL-S, validation tools such as OWL-S Validator [27] and ConsVISor [28] are available.

3.4. Discovery Phase

The discovery phase takes the abstract composite workflow, and finds suitable atomic services for each task in the workflow by querying service repositories.

3.4.1. Approaches

Discovered services can be bound to the corresponding subtasks in the workflow as they are found (design-time binding/static binding) or defined in abstract terms and bound to the subtask at run time (run-time binding/dynamic binding). Design-time binding may not be optimal if the status of the service can change while the process is executing. If services are not bound at discovery, but at run-time, it is possible to find alternative services when necessary.

A query may produce too many results (including unsatisfactory ones) or none at all. Possible reasons are the lack of sufficient description of the web services themselves, too many constraints on the subtasks of the workflow, lack of constraints of the subtask, insufficient decomposition of workflow into suitable subtasks, etc. These issues may require additional tweaking of the workflow or specification.

3.4.2. Tools and Components

The most widely used web service repository is UDDI, an XML-based open-industry initiative [3]. Another common repository is the ebXML registry [29]. These repositories differ in the way information is stored, the type of information stored, and the way they are envisioned to be used [30].

3.5. Execution and Monitoring Phase

The execution and monitoring phase provides deployment and execution of a newly created composite service. This phase includes following control flows that were specified in the workflow and recovery mechanisms to ensure proper execution of composition.

3.5.1. Approaches

If design-time binding was used, selected services have already been bound and the workflow can be executed. However, if run-time binding is used, the discovered services can be checked to verify their availability. Alternate services can be selected if the first choice is unavailable at the time of execution. Postponing binding can lead to a more survivable workflow overall.

Monitoring the execution of the composite service can be thought of as an extension to the validation phase, except that the monitor deals with run-time services. It makes assumptions that the partner services should satisfy (according to the user's specification), and properties that the overall composite service must guarantee assuming that the services behave accordingly. Monitoring tools also examine the input/output messages passed from services as a way to check whether the service conforms to the user-defined interaction constraints and the service's advertised assertions.

When a service is unavailable or fails during the execution, failure recovery is done via web service substitution. When an alternate web service is substituted for a failed one, the new service should support all the functionality of the original service being replaced. In addition, when added to the composition, the new service should not alter the context of the composition [31]. If alternate services are not available, the user may be required to

go back to previous stages to find an alternate candidate workflow, or modify the original workflow to accommodate an alternate service.

3.5.2. Tools and Components

Many execution engines are available for composite services created using WS-BPEL or OWL-S. These engines usually are capable of specification, validation, deployment, and management of composite workflows [26]. Various monitoring tools use AI planning algorithms such as model checking for run-time validation and formulation of properties [6, 24, 32], similar to the available validation tools mentioned in section 3.2.2.

So far we have examined the various phases of web service composition. We examined the possible options for each phase and their pros and cons. A summary of the phases and available options is shown in Figure 4.

Within our discussion of options, we have not mentioned the use of semantics. A semantic framework may require more component and tool support, and add an additional layer of complexity to the process. However, semantic annotations facilitate automatic matchmaking and machine understanding, enabling more accurate specification, matching, and validation. It is an option available throughout the composition process.

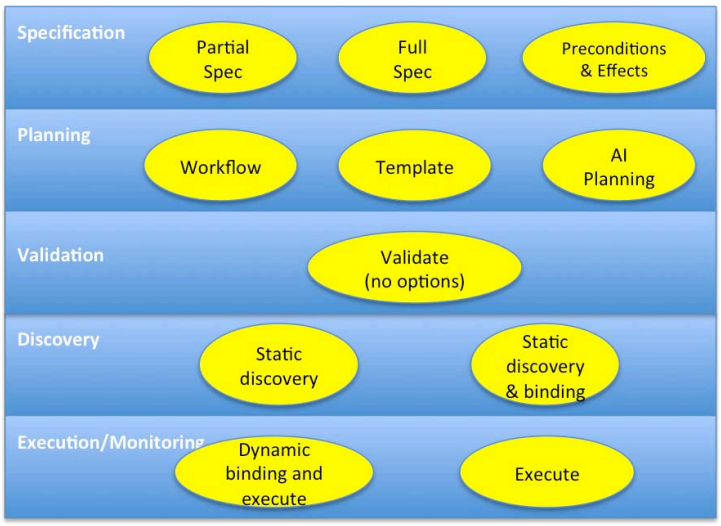

Figure 4. Composition process with options shown for each phase

4. Decision Tree for Web Service Composition

In this section we provide a list of factors that assist in the selection of options in each phase and the overall composition process. The factors are: 1) Availability of templates, 2) User's domain knowledge, and 3) Open-world vs. closed-world assumption.

There are certainly many other factors that may affect the decision-making process, such as the application type, estimated timeframe from specification to deployment, user's level of technical knowledge (such as web service standards), etc. However, these other factors are more orthogonal to the process and therefore are outside the scope of this paper. The factors identified above are described in more detail below.

1) Availability of templates

In our view, templates are a satisfactory compromise between manual composition, where the user retains control over the process, but is laborious, and full automatic composition where the user is freed from the complexity of the process, but must depend on the composition tools to correctly interpret and implement the user's intention. Therefore, barring other factors, the first thing to check is whether templates are available for the particular domain. If so, the user need only create a partial specification that can be used as the search query to retrieve candidate templates. If templates are not available, or if they are available but not satisfactory for the particular task, then alternate methods need to be considered, leading us to the next decision factor.

2) User's domain knowledge

When templates are not available, the user must create one. If the user possesses domain knowledge, he will be able to decompose the task statement and create a full specification that details the specific tasks and data/control flow among the tasks, as well as other non-functional requirements. From this point, the user can easily create an abstract composite workflow in the planning phase, based on the specification he created. However, if the user has limited domain knowledge, it cannot be expected of a user to create a full workflow. Instead, a specification stated in terms of preconditions and effects (P&E) can be created and used to create a workflow using AI planning techniques that does not require knowledge of the process model. Workflows created in either manner can be stored as templates for future use.

3) Open-world vs. closed-world assumption

This is the last factor that we consider in the decision process. Closed-world assumption assumes the environment to be unchanged from the time a plan is generated to the time the plan is executed, disallowing unknown events [4]. Many AI planning algorithms work under this assumption. The open-world assumption is opposite. It provides a way to work under uncertainty while providing flexibility and a degree of execution guarantee. This is especially important when binding services to a workflow. After all, not all advertised or published services may be running and deployed at a given (execution) time. The services may suddenly become unavailable, or may have been redefined by the publisher making it no longer suitable for use in the workflow. Therefore, the assumption made here affects the discovery and execution/monitoring phases. If working under a closed world

assumption, we can discover and bind during design time. However, if working under the open-world assumption, they are defined in abstract terms and matched with available services at run time during the execution stage.

We provide no options and thus no decision factors for the validation phase at this time. The full decision tree is shown in Figure 5 and how it works with the composition framework is depicted in Figure 6.

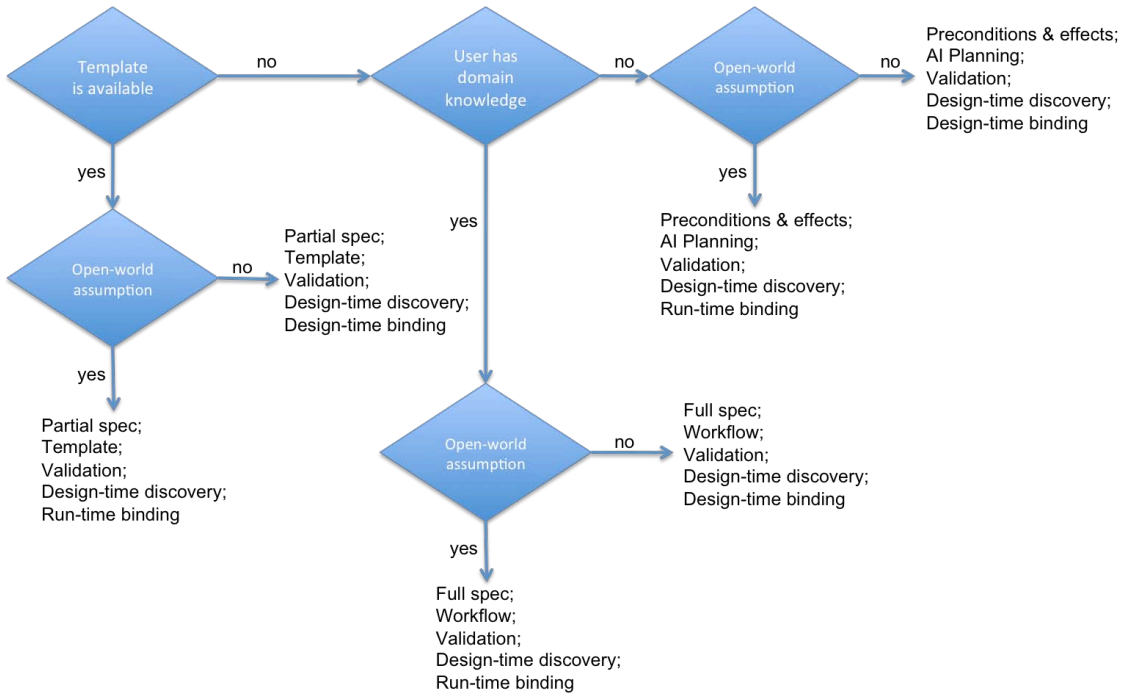

Figure 5. Decision tree for service composition process

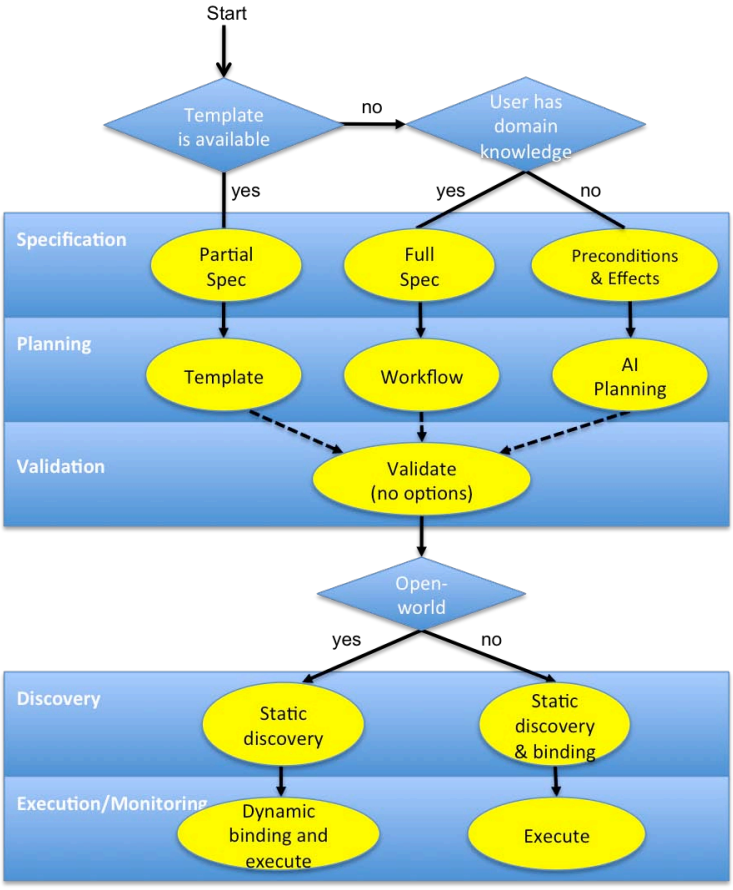

Figure 6. Decision tree within the composition framework

5. Recommendation for the Geospatial Planning Project

The goal of the geospatial planning project is to develop a system for automatically composing individual web services into a composite service, particularly in the geospatial domain. This will provide warfighters with the ability to create accurate mission plans utilizing various geospatial and environmental data sources in a timely manner.

Having outlined some guidelines on how to select options during the composition process, we can now make recommendations for selecting composition options specifically for the geospatial planning project. The summary recommendations are as follows. An explanation behind the reasoning of the selection of these options will follow.

- Specification Phase: Partial specification
- Planning Phase: Template-based approach
- Validation Phase: Syntactic and semantic validation of template
- Discovery Phase: Static discovery with delayed binding
- Execution/Monitoring Phase: Run-time binding

Figure 7 shows the composition decision path for the geospatial planning project with options highlighted. Before we introduce the reasoning behind these recommendations, one thing to note is that we decided to use semantics throughout the composition process. This requires that all components and tools utilize semantics (via ontologies). To create an environment in which semantics is used, domain experts will have to create various ontologies that can be used to semantically annotate SOA components. Web service publishers/authors will have to annotate their services (descriptions and interfaces) with these ontologies. Furthermore, widely used (state of the art) tools need to be modified/extended so that they can process the semantics available. This includes tools that are available for every phase in the composition process. Hence, the following subsection will touch on the issues of incorporating semantics as well.

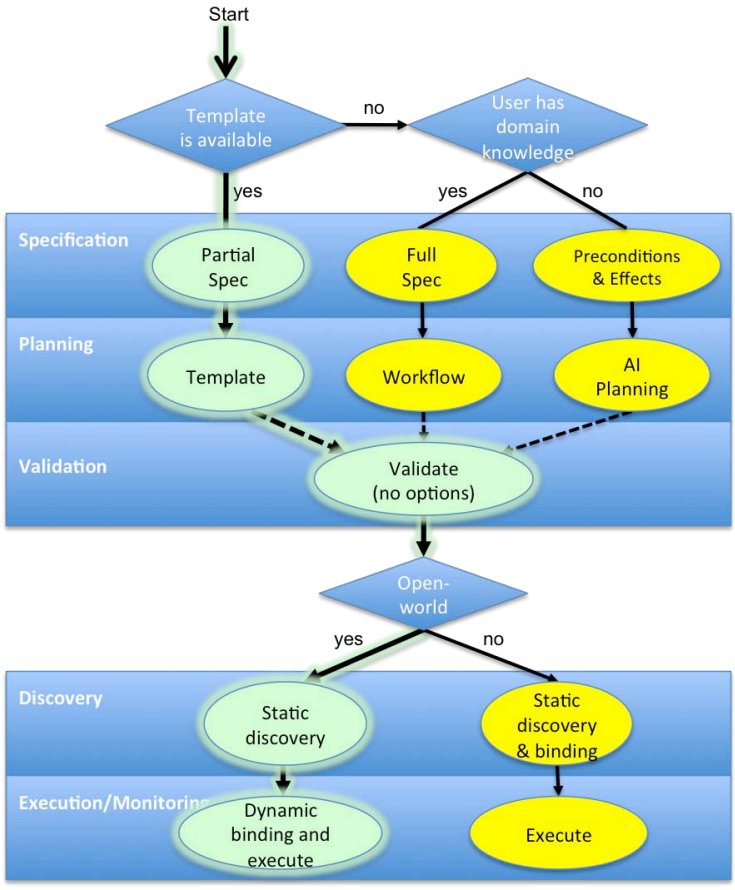

Figure 7. Composition decision path for the Geospatial Planning Project

5.1. Reasons for the Recommendation

For the specification phase, we chose a partial specification approach. The geospatial domain is broad, encompassing many areas such as bathysphere, telemetry, imaging, etc. Therefore, it is hard to expect one user to be sufficiently familiar with all these domains, as well as the non-functional aspects of the required task. In addition, we believe that the use of templates in the planning stage is a good tradeoff between manual and fully

automatic composition. Therefore, the specification phase need not require a full-blown specification. At a minimum, the specification should state the initial IO parameters, and the task goal as well as any constraints the user may require (e.g. the output must be in a certain format) to aid the search. The specification can be created using OWL-S Profile with semantic annotations. An example tool that can be used is the OWLS-Editor with a graphical interface to create the control structure.

For the planning phase, we choose a template-based approach (assuming templates are available) so that the partial specification from the previous phase can be used to search the repository (UDDI) for matching templates[1]. The semantic annotations used in the specification can assist in searching for more precise matches. Once a suitable template is found, it can then be customized with user's constraints. If no templates exist for the particular task, we propose/suggest that the user create his own workflow (which certainly could be saved as a template), from existing templates that can make up parts of the complex task, and adding components where necessary, or creating his own workflow from scratch using available tools such as OWL-S editor. Other arguments for using a template-based approach are the reusability factor and the minimum level of time and effort involved in composition.

For the validation phase, there is only one option to choose from. We will just validate the abstract process in terms of the validity of the workflow itself (i.e. syntactic validation), and to check if the workflow satisfies the goals (both functional and non-functional) defined by the user in the specification phase (i.e. semantic validation). Several tools are available for reasoning and validation of OWL-S. OWL-S Reasoner, Pellet, OWL-S Validator, and ConsVISor are example tools. We need to determine which of these tools would best suit our needs.

For the discovery phase, we adopt an open-world assumption since we cannot guarantee that discovered services will be available or unchanged at a later time. We propose to search for candidate atomic services to populate the abstract workflow, but defer binding the services to the workflow at a later time. When more than one candidate service is found for a specific task, we can keep the list of candidates sorted according to the user's stated preferences (regarding QoS, execution time, reliability, etc). The open world assumption allows us to expect that services up and running at discovery time not being available at run time, providing more flexibility within the execution stage. We can use NRL's semantic UDDI [33] as a repository to store descriptions of both atomic services and templates.

For the execution and monitoring phase, the effects of the open-world assumption in the discovery phase trickle down to enable run-time binding of atomic services to tasks in the abstract workflow. Services are bound to the workflow as it is executed, based on the results of the discovery phase and current state of the service. There are OWL-S execution tools such as OWL-S VM that provide a generic processor for the OWL-S process model and automatically invoke OWL-S services (this is an OWL-S execution

[1] While this phase does search for templates (which themselves can be thought of as a web service), and thus requires access to a repository, it is differentiated from the subsequent discovery phase.

environment). For monitoring, one possible tool we can employ is a network monitoring tool we developed at NRL. It is a network management tool that allows the user to only monitor the parts of the system that affect the services he is using. It can manage the candidate list of atomic services discovered in the previous phase and through a selector plug-in, determine service selection priority. Thus, if a service becomes unavailable, the tool selects the next service available in the list, sorted according to some predefined user ranking (e.g. QoS parameter).

5.2. Components Required to Realize Geospatial Planning Project

We will need a variety of tools and components to realize the geospatial planning project. While some required tools and components are available, some may not be sufficient for our purposes, necessitating additional research. This section examines the issues we need to address to implement the geospatial planning project.

First, since we will be utilizing semantics throughout the project, we will need to have a way to semantically annotate the various web service components. We require various geospatial related domain ontologies, and other ontologies dealing with the non-functional aspects of the tasks. We expect to be able to rely partially on previous or related work such as the NRL Security Ontology [34]. There are three major metadata specifications related to geographic digital data: Geographic Metadata Standard ISO 19115:2003 [20], Dublin Core Metadata Initiative, and the Content Standard for Digital Geospatial Metadata, published by the Federal Geographic Data Committee (cite OntoMet). There is also the ISO 19115 ontology[2] for Geographic information. We need to examine these and see if it any are suitable for our needs or can be extended. We also need to identity other annotations we will want to use with our services to determine if there are already ontologies that provide the required semantics. If not, we will need to create them. We will use OWL, the Web Ontology Language [cite] for this purpose.

Second, we need to create a test bed of atomic services that can be used in the orchestration of a more complex service. These services may initially be created with WSDL, but should also be semantically annotated using OWL-S. Since we propose using a template-based approach for composition, we will need to research various template-related issues. For example, we will need to create a test bed of templates also annotated with OWL-S. Both the atomic services and templates need to be published into the semantic UDDI. This requires examining how to store templates into UDDI, and what degree of specificity should be added to the templates. In other words, we need to research how much of the functional and non-functional aspects of a task should be included in a generic template, and how much of it should be customizable.

---

[2] ISO 19115 "Geographic Information - Metadata"[1] is a standard of the International Organization for Standardization (ISO). It is a component of the series of ISO 191xx standards for Geospatial metadata. ISO 19115 defines how to describe geographical information and associated services, including contents, spatial-temporal purchases, data quality, access and rights to use.

Lastly, we will also need to consider what aspects of the task will be validated, and how best to model the process logic to facilitate validation.

While we have suggested some tools to use for some parts of the composition process, we need to examine these and see if they truly meet our goals. And for others, like the validation phase, we need to decide upon a tool. We also need to see if these various toolsets work together seamlessly throughout the composition process from specification all the way to execution and monitoring.

6. Conclusion

Manual web service composition is an error-prone, tedious process. On the other hand, while automatic web service composition frees the user from the burdens of manual composition, it takes control from the user. Also, full automatic composition is not at the technical stage yet where the user can fully trust the composition and execution results to be what the user intended it to be. For real-world applicability, we need to find an acceptable middle ground between manual and automatic composition that frees the user from the complexity of the process, but at the same time gives the user enough involvement in the process that provides them with confidence of the end result.

While there are many approaches to web service composition, there is no one-solution-fits-all approach. There is no absolute guideline as to how to proceed in the composition process. The decision to follow a certain composition path within the framework depends on many factors, the salient ones outlined and discussed here and examined within the scope of the geospatial planning project.

7. References

1. OASIS Web Services Business Process Execution Language (WSBPEL) TC, *Web Services Business Process Execution Language Version 2.0*. 2007: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.
2. Martin, D., et al., *OWL-S: Semantic Markup for Web Services*. 2004: http://www.w3.org/Submission/OWL-S/.
3. UDDI Spec Technical Committee, *UDDI Version 3.0.2*. 2004: http://www.uddi.org.
4. Agarwal, V., et al., *Understanding Approaches for Web Service Composition and Execution*, in *COMPUTE 2008*. 2008: Bangalore, India.
5. Alamri, A., M. Eid, and A.E. Saddik, *Classification of the state-of-the-art dynamic web services composition techniques.* International Journal of Web and Grid Services, 2006. **2**(2): p. 148-166.
6. Fu, Y., Z. Dong, and X. He, *Modeling, validating and automating composition of web services*, in *Proceedings of the 6th international conference on Web engineering*. 2006: Palo Alto, California, USA p. 217-224.
7. Marconi, A., M. Pistore, and P. Traverso, *Automated Composition of Web Services: the ASTRO Approach.* IEEE Data Engineering Bulletin, 2008. **31**(3): p. 23-26.
8. Rao, J. and X. Su, *A Survey of Automated Web Service Composition Methods*, in *SWSWPC 2004*. 2004: San Diego, CA, USA.

9. Svatek, V. and M. Vacura, *Automatic Composition of Web Analysis Tools: Simulation on Classification Templates*, in *First International Workshop on Representation and Analysis of Web Space (RAWS-05)*. 2005: Prague-Tocna.
10. Wu, Z., et al., *Automatic Composition of Semantic Web Services using Process and Data Mediation*. 2007, Kno.e.sis Center, Wright State University: Dayton, Ohio, USA.
11. Geebelen, K., S. Michiels, and W. Joosen, *Dynamic reconfiguration using template based web service composition*, in *3rd workshop on Middleware for service oriented computing (MW4SOC '08)*. 2008, ACM: Leuven, Belgium.
12. Guo, J., et al., *A Template-Based Orchestration Framework for Hybrid Services*, in *2008 Fourth Advanced International Conference on Telecommunications (AICT)*. 2008: Athens, Greece. p. 315-320.
13. Sirin, E., B. Parsia, and J. Hendler, *Template-based Composition of Semantic Web Services*, in *AAAI Fall Symposium on Agents and the Semantic Web*. 2005: Virginia, USA.
14. Peer, J., *Web Service Composition as AI Planning - a Survey*. 2005, University of St. Gallen, Switzerland.
15. Bartalos, P. and M. Bieliková, *Enhancing Semantic Web Services Composition with User Interaction*, in *2008 IEEE International Conference on Services Computing (SCC 2008)*. 2008: Honolulu, Hawaii, USA. p. 503-506.
16. Kim, J. and Y. Gil, *Incorporating tutoring principles into interactive knowledge acquisition.* International Journal of Man-Machine Studies, 2007. **65**(10): p. 852-872.
17. Arpinar, I.B., et al., *Ontology-driven Web Services Composition Platform*, in *IEEE International Conference on E-Commerce Technology (CEC 04)*. 2004: San Diego, California, USA. p. 146-152.
18. Cardoso, J. and A.P. Sheth, *Semantic Web Services and Web Process Composition*, in *First International Workshop, SWSWPC 2004*. 2004: San Diego, CA, USA.
19. Yue, P., et al., *Semantics-based automatic composition of geospatial Web service chains.* Computers and Geosciences, 2007. **33**(5): p. 649-665.
20. Yue, P., et al. *Semantics-enabled Metadata Generation, Tracking and Validation in Geospatial Web Service Composition for Mining Distributed Images* in *Proceedings of the 2007 IEEE International Geoscience and Remote Sensing Symposium (IGARSS07)*. 2007. Barcelona, Spain.
21. Agarwal, S., *A Goal Specification Language for Automated Discovery and Composition of Web Services*, in *Proceedings of the IEEE/WIC/ACM International Conference on Web Intelligence (WI 07)*. 2007, IEEE Computer Society: Silicon Valley, USA. p. 528--534.
22. Pathak, J., et al., *MoSCoE: A Framework for Modeling Web Service Composition and Execution*, in *22nd International Conference on Data Engineering Workshops*. 2006, IEEE Computer Society.
23. Jie, G., et al., *A Template-Based Orchestration Framework for Hybrid Services*, in *2008 Fourth Advanced International Conference on Telecommunications (AICT)*. 2008: Athens, Greece. p. 315-320.

24. Bianculli, D., C. Ghezzi, and P. Spoletini. *A Model Checking Approach to Verify BPEL4WS Workflows* in *International Conference on Service-Oriented Computing and Applications (SOCA 07)*. 2007. Newport Beach, California, USA.
25. Eclipse, *BPEL Project Home*: http://www.eclipse.org/bpel/.
26. Oracle, *BPEL Process Manager*: http://www.oracle.com/technology/products/ias/bpel/index.html.
27. Mindswap.org, *OWL-S Validator*: http://www.mindswap.org/2004/owl-s/validator/.
28. Versatile Information Systems, *ConsVisor Main Page*: http://vistology.com/consvisor.
29. OASIS Technical Committee, *ebXML Technical Architecture Specification v1.04*. 2001: http://www.ebxml.org/.
30. Sun Microsystems, *Effective SOA Deployment using an SOA Registry Repository*, in *White paper*. 2005, Sun Muicrosystems: http://www.sun.com/products/soa/registry/soa_registry_wp.pdf.
31. Pathak, J., S. Basu, and V. Honavar, *On Context-Specific Substitutability of Web Services* in *5th IEEE Intl. Conference on Web Services (ICWS 2007)*. 2007: Salt Lake City, UT, USA.
32. Sadiq, S., et al., *Data Flow and Validation in Workflow Modelling*, in *Conferences in Research and Practice in Information Technology (ADC 2004)*. 2003: Dunedin, New Zealand.
33. Luo, J., et al., *Adding OWL-S Support to the Existing UDDI Infrastructure*, in *2006 IEEE International Conference on Web Services (ICWS 2006)*. 2006: Chicago, Illinois, USA.
34. Kim, A., J. Luo, and M. Kang, *Security Ontology to Facilitate Web Service Description and Discovery*. Journal of Data Semantics, 2007. **9**: p. 167-195.